Exact Bayesian bin classification: A fast alternative to Bayesian classification and its application to neural response analysis

J Comput Neurosci (2008) 24:21–35
DOI 10.1007/s10827-007-0039-5

Exact Bayesian bin classification: a fast alternative to Bayesian classification and its application to neural response analysis

D. Endres · P. Földiák

Received: 6 May 2005 / Revised: 6 November 2006 / Accepted: 16 April 2007 / Published online: 14 June 2007
© Springer Science + Business Media, LLC 2007

Abstract We investigate the general problem of signal classification and, in particular, that of assigning stimulus labels to neural spike trains recorded from single cortical neurons. Finding efficient ways of classifying neural responses is especially important in experiments involving rapid presentation of stimuli. We introduce a fast, exact alternative to Bayesian classification. Instead of estimating the class-conditional densities $p(x|y)$ (where x is a scalar function of the feature[s], y the class label) and converting them to $P(y|x)$ via Bayes' theorem, this probability is evaluated directly and without the need for approximations. This is achieved by integrating over all possible binnings of x with an upper limit on the number of bins. Computational time is quadratic in both the number of observed data points and the number of bins. The algorithm also allows for the computation of feedback signals, which can be used as input to subsequent stages of inference, e.g. neural network training. Responses of single neurons from high-level visual cortex (area STSa) to rapid sequences of complex visual stimuli are analysed. Information latency and response duration increase nonlinearly with presentation duration, suggesting that neural processing speeds adapt to presentation speeds.

Action Editor: Alexander Borst

D. Endres (✉) · P. Földiák
School of Psychology, University of St. Andrews,
St Andrews KY16 9JP, UK
e-mail: dme2@st-andrews.ac.uk

P. Földiák
e-mail: Peter.Foldiak@st-andrews.ac.uk

Keywords Classification · Exact Bayesian inference · Neural decoding

1 Introduction

Bayesian classification is a widely used method in many fields of scientific inference and engineering. In this paper we develop a new classification method and demonstrate its usefulness for analysing signals recorded from single cortical neurons. Classification in this context involves selecting a time window. The neural signal from this window is used for identifying the stimulus that evoked the neural response. This identification is necessary for the interpretation of the function of the investigated neuron by the experimenters.

Traditionally, individual stimuli were presented well isolated in time allowing easy separation of the responses. The recently developed methods of applying Rapid Serial Visual Presentation to single-cell neurophysiological experiments (Földiák et al. 2004; Keysers et al. 2001) allow a more efficient use of the limited experimental time, however, they made the task of response separation significantly harder. The selection of an optimal time window is more critical in these experiments. This paper presents a method that is well suited for selecting such optimal windows for stimulus classification.

Suppose we wanted to determine whether an object O_k belonged to any one of C classes. To do so, we observe a vector of features $\vec{w}_k$ (which is in the following assumed to be a vector of real numbers), which we hope to contain information necessary to assign a class label y_k to the object in question. Usually, due to noise or incompleteness of the available information, this cannot be done with certainty. Hence, we try to estimate the probability $P(y_k|\vec{w}_k)$ that the object belongs to class $y_k \in \{1, \ldots, C\}$. A common method—Bayesian classification—estimates the class-conditional densities $p(\vec{w}|y, \vec{\theta}_y)$ ($\vec{\theta}_y$ are the parameters of the density model for class y) and then uses Bayes' theorem to predict

$$P(y|\vec{w}, \vec{\theta}_1, \ldots, \vec{\theta}_C) = \frac{p(\vec{w}|y, \vec{\theta}_y)P(y)}{\sum_{y'} p(\vec{w}|y', \vec{\theta}_{y'})P(y')} \tag{1}$$

where $P(y)$ is the probability of O_k belonging to class y prior to observing $\vec{w}$.

The correct way—from a Bayesian perspective—to do away with the dependency of the l.h.s. on the $\vec{\theta}_y$, is to integrate them out of the prediction:

$$P(y|\vec{w}, D) = \int_{\vec{\theta}_1} d\vec{\theta}_1 \ldots \int_{\vec{\theta}_C} d\vec{\theta}_C P(y|\vec{w}, \vec{\theta}_1, \ldots, \vec{\theta}_C) p(\vec{\theta}_1, \ldots, \vec{\theta}_C|D) \tag{2}$$

$p(\vec{\theta}_1, \ldots, \vec{\theta}_C|D)$ is probability density of the $\vec{\theta}_y$ given previously observed data D [comprised of pairs $(\vec{w}_k, y_k)$] and any other available prior information regarding them (the implicit dependency on the model class is omitted here and in the following).

Performing integrals of this type is generally very difficult. Therefore, a variety of approximation methods for their evaluation have been derived in the past, e.g. MAP (maximum a posteriori), ML (maximum likelihood, basically MAP with a uniform prior) , Laplace approximation (where the posterior is approximated by a Gaussian centred at its maximum; Gelman et al. 1995), Monte-Carlo simulations (i.e. inspired sampling; Neal 1996) or variational techniques (Jordan et al. 1999). In the following, we will show how the problem can be circumvented (at leat in part) by directly inferring $P(y|\vec{w})$ from the data.

2 A simple model for $P(y|\vec{w})$

Assume one had a multiset of K labelled feature vectors $D = \{(\vec{w}_k, y_k)\}$. The classification task can then be decomposed into two steps:

Step 1. Find a suitable mapping $f(\vec{w}) \mapsto x, x \in [0, 1]$, such that x contains all the information from $\vec{w}$ that pertains to the classification,

Step 2. Infer the probabilities $P(y|x, D_f)$, where $D_f = \{(x_k, y_k)\}$ are the data after the $\vec{w}_k$ have been mapped onto the x_k through $f(.)$.

While step 1 is by no means trivial, the following arguments will focus mostly on step 2. It should be noted, however, that step 1 is always possible (to any given degree of accuracy): the most general information which $\vec{w}$ can contain about y is given by the probability distribution $P(y|\vec{w})$. For $C = 2$, choose $f(\vec{w}) \mapsto x$ such that $P(y = 1|x) = x$. For $C = 3$, the possible probability distributions can be represented by points inside a 2 dimensional simplex (i.e. a triangle). If the desired accuracy is ϵ, then cover the triangle by smaller triangles of width and height $< \epsilon$, enumerate those small triangles in some fashion (e.g. such that the probabilities change as smoothly as possible between successive triangles) and map this enumeration in an order-preserving way onto x. Each possible value of x then represents a probability distribution. The generalisation of this procedure to any finite C is straightforward. While it may not be the best way of performing $f(\vec{w}) \mapsto x$, it serves as an indication that this mapping is always possible for a given ϵ. Ways of finding $f(.)$ are discussed in Section 10.1.

Since we will present an exact solution for step 2, the BBCa (Bayesian bin classification algorithm), any approximation error in the classification that might be introduced through step 1 will be kept at a minimum. Furthermore, in the decomposition suggested here, $f(.)$ can possibly be modelled in a simpler manner because it no longer needs to deal with the mapping onto the class labels.

2.1 A bin model for $P(y|x)$

To carry out step 2, the following model will be employed:

Divide the interval [0, 1] by M points z_j, so as to get $M + 1$ bins (see Fig. 1). Within each bin, the probability $P(y|x)$ is assumed to be constant w.r.t. x. There are C such probabilities per bin, denoted by $c_y^j = P(y|x \in (z_{j-1}, z_j])$, where y is the class and j is the bin (bin j is the interval between z_{j-1} and z_j, bin 0 is the interval

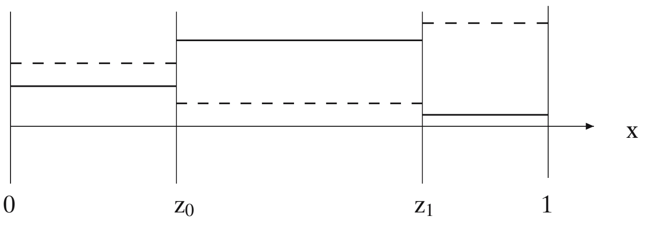

Fig. 1 An example model. The interval between 0 and 1 is divided into three bins at points z_0 and z_1. Within each bin $j \in \{0, 1, 2\}$, x is mapped onto $P(y|x) = c_y^j$, which are constant within the bin. Two classes are possible, their probabilities are indicated by the *solid and dashed lines*

between 0 and z_0, hence bin M is the interval between z_{M-1} and 1). This is not as restrictive as it may seem at first, since it is possible to approximate any continuous function with arbitrary accuracy by a piecewise constant function, given that the number of bins is not limited.[1]

The c_y^j are normalised with respect to the classes within their bin, i.e.

$$\forall j \in \{0, \ldots, M\} : \sum_y c_y^j = 1$$

$\{c_y^j\}$ denotes the set of these x-conditional probabilities in bin j.

For a given configuration of M,$\{z_j\}$ and $\{\{c_y^j\}\}$, the probability of the dataset then becomes (assuming that the data points have been drawn independently of each other):

$$P(D|\{\{c_y^j\}\}, \{z_j\}, M) = \prod_k c_{y_k}^{j_k} \tag{3}$$

where j_k is the bin that contains $x_k = f(\vec{w}_k)$. Once observed, one can easily reorder the data points so that $x_k < x_{k+1}$, i.e. the data is ordered according to x_k, which will be assumed from here on.

Classifying a new feature vector $\vec{w}'$ involves evaluating

$$P(y'|x', D_f) = \frac{P(D'_f)}{P(D_f)} \tag{4}$$

where D'_f is the data(multi)set with $(x' = f(\vec{w}'), y')$ added to it, i.e. $D'_f = D_f \cup \{(x', y')\}$. Assuming the $\{\{c_y^j\}\}$ and the $\{z_j\}$ have already been marginalised, Eq. (4) is equivalent to

$$P(y'|x', D_f) = \frac{\sum_M P(D'_f|M)P(M)}{\sum_M P(D_f|M)P(M)} \tag{5}$$

where $P(M)$ is the prior probability for a model with M bin boundaries in $[0, 1]$. The sums run over all values of M which one chooses to include into the calculation. This choice could be made (in the case of uniform $P(M)$) by selecting those M that have a high enough evidence $P(D_f|M)$ to contribute significantly to Eq. (4). As y' is the quantity to be determined, one has to evaluate Eq. (4) for all possible values of y' and then pick the one with the highest probability, to minimise the chance of misclassification.

Thus, one needs to compute $P(D_f|M)$, i.e. the evidence for a model with M bin boundaries:

$$\begin{aligned} P(D_f|M) &= \int d\{z_j\} \int d\{\{c_y^j\}\}\ p(D_f, \{\{c_y^j\}\}, \{z_j\}|M) \\ &= \int d\{z_j\} \int d\{\{c_y^j\}\}\ P(D_f|\{\{c_y^j\}\}, \{z_j\}, M) \\ &\quad \times p(\{\{c_y^j\}\}|\{z_j\}, M)p(\{z_j\}|M) \end{aligned} \tag{6}$$

where

$$\int d\{z_j\} := \int_0^1 dz_{M-1} \int_0^{z_{M-1}} dz_{M-2} \ldots \int_0^{z_1} dz_0 \tag{7}$$

$$\int d\{c_y^j\} := \int_0^1 dc_0^j \int_0^1 dc_1^j \ldots \int_0^1 dc_{C-1}^j \delta(1-\sum_{y=0}^{C-1} c_y^j) \tag{8}$$

$$\int d\{\{c_y^j\}\} := \int d\{c_y^0\} \int d\{c_y^1\} \ldots \int d\{c_y^M\} \tag{9}$$

The way the integration boundaries are chosen in Eq. (7) ensures that the ordering of the z_j is maintained, and the Dirac-delta function in Eq. (8) enforces normalisation of the probabilities in bin j.

It is also possible, in a similar fashion, to determine confidence intervals on the predicted probabilities, via

$$\mathrm{Var}\left[P(y'|x', D_f, M)\right] = \frac{P(D''_f|M)}{P(D_f|M)} - \left(\frac{P(D'_f|M)}{P(D_f|M)}\right)^2 \tag{10}$$

where D''_f is the data(multi)set with the point (x', y') added twice.[2]

Since $p(\{\{c_y^j\}\}|\{z_j\}, M)$ and $p(\{z_j\}|M)$ do not depend on the data D_f, they are prior densities which will be assigned in the following way:

- $p(\{z_j\}|M)$ depends on M only insofar as $0 \le j < M$. Furthermore, it will be assumed that $z_j \le z_{j+1}$, i.e. the z_j are ordered, but otherwise no preferences for their locations in the interval $[0, 1]$ will be expressed in the prior. It will also be assumed that, apart from the ordering, they are independent of each other. The prior thus becomes $p(\{z_j\}|M) = \prod_j p(z_j)$, where $p(z_j) = p_z = \text{const} > 0$ if $z_j \le z_{j+1}$ and 0 otherwise.

[1]Define $f_n(x) \mapsto f(\frac{\xi}{n})$, where ξ is the greatest integer smaller than $x \cdot n$. Since f is continuous, $\lim_{n\to\infty} f_n(x) = f(x)$.

[2]$\mathrm{Var}\left[(P(y'|x', D_f, M))\right] = E\left[P(y'|x')^2\right] - E\left[P(y'|x')\right]^2$, where the expectations are w.r.t. the posterior of $P(y'|x')$. This posterior is given by the integrand of Eq. (6) divided by Eq. (6). To compute the expectation of $P(y'|x')$, multiply the posterior by $c_{y'}^{j_{x'}}$, where $j_{x'}$ is the bin containing x' and integrate. By virtue of Eq. (3), this is computationally equivalent to adding the point (x', y') to the data(multi)set. Likewise, to compute the expectation of the square of $P(y'|x')$, add this point twice, thus yielding the expectation of $\left(c_{y'}^{j_{x'}}\right)^2$.

- $p(\{\{c_y^j\}\}|\{z_j\}, M) = \prod_j \prod_y p(c_y^j)$, i.e. the prior densities of the x-conditional probabilities in bin j are independent of the locations of the bin boundaries z_j and independent of each other, except for the constraint that the c_y^j be normalised w.r.t. the classes. Hence, it will be assumed that the $p(c_y^j) = p_c$ are constant and equal (subject to the normalisation constraint).

3 The Bayesian bin classification algorithm

3.1 Computing $p(\{\{c_y^j\}\}|\{z_j\}, M)$

The prior $p(\{\{c_y^j\}\}|\{z_j\}, M)$ has to be normalised w.r.t. $\{\{c_y^j\}\}$, i.e.

$$\int d\{\{c_y^j\}\} p(\{\{c_y^j\}\}|\{z_j\}, M) = 1 \tag{11}$$

Since the sets $\{c_y^j\}$ are assumed to be independent of each other, the integration over each set can be performed independently of the others. In the following, the superscript j (which denotes the bin) will therefore be dropped. As $0 \leq c_y \leq 1$, one needs to compute integrals of the form

$$I(l_0, \ldots, l_{C-1}) = \int_0^1 dc_0 (c_0)^{l_0} \ldots \int_0^1 dc_{C-1} (c_{C-1})^{l_{C-1}} \times \\ \times p_c^C \delta\left(1 - \sum_{y=0}^{C} c_y\right) \tag{12}$$

where l_c means the number of data points which belong to class c, and the δ-function ensures that the set of c_y are always a probability distribution. This yields the normalisation constant of a Dirichlet distribution (see e.g. Hutter 2002)

$$I(l_0, \ldots, l_{C-1}) = p_c^C \frac{\prod_y l_y!}{(\sum_y l_y + C - 1)!} \tag{13}$$

The integral over the prior then becomes:

$$\int d\{\{c_y^j\}\} p(\{\{c_y^j\}\}|\{z_j\}, M) = (I(0, \ldots, 0) p_c^C)^{M+1} \tag{14}$$

Hence, $p_c = (C-1)!^{\frac{1}{C}}$ and the prior is

$$p(\{\{c_y^j\}\}|\{z_j\}, M) = (C-1)!^{M+1} \tag{15}$$

4 Computing $p(\{z_j\}|M)$

This prior is subject to the normalisation constraint

$$\int d\{z_j\} p(\{z_j\}|M) = 1 \tag{16}$$

Since $p(\{z_j\}|M)$ is assumed to be constant,

$$p(\{z_j\}|M) = M! \tag{17}$$

5 Computing the evidence

Now the evaluation of Eq. (6) can be continued. Assume that one chose a particular binning of the interval [0, 1], represented by the set $\{z_j\}$. It then becomes possible to carry out the integrations over the $\{\{c_y^j\}\}$:

$$p(D_f|\{z_j\}, M) = \int d\{\{c_y^j\}\} \\ \times \underbrace{P(D_f|\{\{c_y^j\}\}, \{z_j\}, M)}_{\text{Eq. (3)}} \underbrace{p(\{\{c_y^j\}\}|\{z_j\}, M)}_{\text{Eq. (15)}} \tag{18}$$

Due to the assumed form of the prior and of the model, this probability is a product, consisting of one factor for each bin. Each factor is of the same form as Eq. (13), the exponents of each c_n being the number of data points (in the following denoted by l_n^j) belonging to class n in the bin j, i.e. the posterior of the c_n is a Dirichlet distribution . Hence one finds

$$P(D_f|\{z_j\}, M) = \prod_{j=0}^{M} \underbrace{I(l_0^j, \ldots, l_{C-1}^j)}_{I_{k_1,k_2}} \tag{19}$$

where x_{k_1}, ..., x_{k_2-1} are the data points in bin j.

What remains is the integration over all possible configurations of bins:

$$P(D_f|M) = \int d\{z_j\} P(D_f|\{z_j\}, M) \underbrace{p\left(\{z_j\}|M\right)}_{\text{Eq. (17)}} \tag{20}$$

5.1 Integrating over $\{z_j\}$

Note that the integrand of Eq. (20) is constant as long as the z_j move between data points. Only when a bin boundary crosses over a data point does its value change. Hence, this integral can be written as a sum. Each summand is the product of the integrand and the total volume occupied by the z_j for that value of the integrand. Let

$$\hat{x}_{k,m} := \frac{(x_k - x_{k-1})^m}{m!} \tag{21}$$

with $x_{-1} = 0$ and $x_K = 1$. If the z_j are distributed in such a way that at most one z_j lies between two adjacent data points, then the total volume occupied by this configuration is the product of the differences between the two data points enclosing z_j, i.e. the product of the corresponding $\hat{x}_{k,1}$. In case there are several of the z_j between the same two adjacent data points, the contribution to the volume will be $\hat{x}_{k,m}$, where m is the number of z_j found in this interval. This follows directly from the ordering constraint imposed upon the z_j.

Now assume $M = 1$. Equation (20) then becomes:

$$P(D_f|M=1) = M! \cdot \sum_{k=0}^{K} I_{0,k} \underbrace{\hat{x}_{k,1} I_{k,K}}_{\hat{E}_{k,1}} \tag{22}$$

For $M = 2$, it would be:

$$P(D_f|M=2) = M! \cdot \sum_{k=0}^{K} I_{0,k} \underbrace{\hat{x}_{k,2} I_{k,K}}_{\hat{E}_{k,2}} + M! \cdot \sum_{k=0}^{K-1} I_{0,k} \underbrace{\hat{x}_{k,1} \sum_{\kappa=k+1}^{K} I_{k,\kappa} \underbrace{I_{\kappa,K} \hat{x}_{\kappa,1}}_{\hat{E}_{\kappa,1}}}_{\hat{E}^{\text{new}}_{k,2}=\alpha \hat{x}_{k,1}} . \tag{23}$$

$\hat{E}_{k,m}$ is the sub-evidence of a sub-model with m bin boundaries (not counting those at 0 and 1) which includes only the points $x_i \geq x_{k-1}$, and α is the sum over all sub-models from the previous M which include only points to the right of x_k. Therefore, when $M = 2$, the evidence for $M = 1$ can be evaluated as well with little extra computational overhead. Furthermore, while computing $P(D_f|M = 1)$, one also evaluates the first part of $P(D_f|M = 2)$, i.e. those configurations where both z_j lie between the same two data points. Then the array $\hat{E}_{k,2}$ can be reused for the computation of the second part, where the z_j are between different pairs of data points. One can proceed in this fashion until the desired M is reached (see Fig. 2). More generally: when evaluating the contribution of a sub-model which has m_0 bin boundaries between x_k and x_{k+1}, and m bin boundaries to the right of x_{k+1}, then one can reuse α to compute the contributions of all sub-models which have $m_1 > m_0$ bin boundaries between x_k and x_{k+1}, because they differ only by a factor $\hat{x}_{k,m_0}$ (for m_0 boundaries) versus $\hat{x}_{k,m_1}$ (for m_1 boundaries).

Should $M > K$, then it is not possible to construct submodels which have at least one data point between adjacent z_j. Thus, the iteration over M can be stopped earlier. In pseudo-code:

```
1. For $k:=0$ to $K$, $m:=1$ to $M$: compute $\hat{E}_{k,m}:=I_{k,K}\hat{x}_{k,m}$
2. For $m:=1$ to $M$: initialise $E_m := \sum_{k=0}^{K} I_{0,k}\hat{E}_{k,m}$
3. For $m:=2$ to $\min(M, K+1)$, $k:=0$ to $K+1-m$:
   a. For $\mu:=m$ to $M$: reset $\hat{E}_{k,\mu}:=0$
   b. For $\mu:=m$ to $M$
      i. Compute $\alpha := \sum_{\kappa=k+1}^{K+2-m} \hat{E}_{\kappa,\mu-1} I_{k,\kappa}$
      ii. For $\nu:=\mu$ to $M$: add $\alpha\hat{x}_{k,\nu-\mu+1}$ to $\hat{E}_{k,\nu}$
   c. For $\mu:=m$ to $M$: add $I_{0,k}\hat{E}_{k,\mu}$ to $E_\mu$
4. return $E_m$
```

This yields $P(D_f|m) = E_m \cdot m!$ for all $1 \leq m \leq M$ (the factor $m!$ is due to the prior over the $\{z_j\}$, see Eqs. (23) and (17). A close look at step 3(b)i reveals that the computational complexity is $O(K^2M^2)$, because one expects $M < K$ for real world applications, or even $M \ll K$.

This way of organising the calculation is computationally similar to the sum-product algorithm for factor graphs (Kschischang et al. 2001). The 'messages' passed on from one m-level to the next are $\hat{E}_{\kappa,\mu-1}$, whereas within one level, a sum over all the 'messages' from the previous level is performed.

6 Comparison to other classification methods

The performances of three other classification methods, support vector machines (SVM), Gaussian process

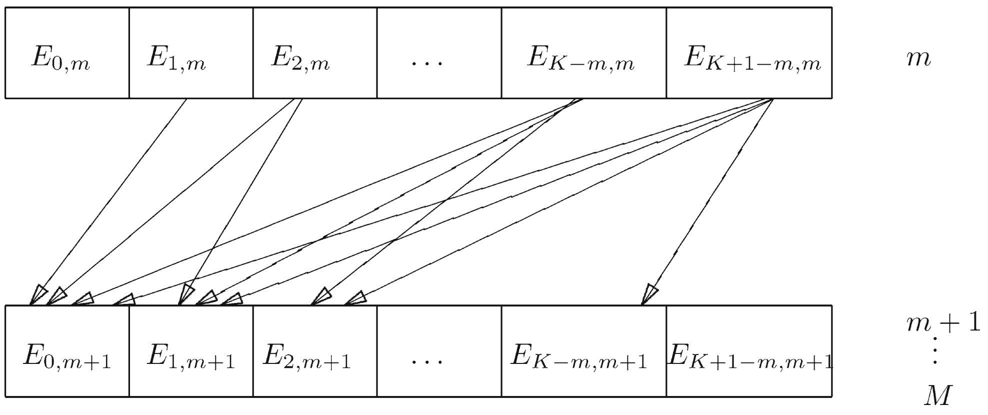

Fig. 2 When evaluating the evidence contribution of sub-models with $m+1$ bin boundaries, one of which is found between x_k and x_{k+1}, then all contributions of models with more than one bin boundary between x_k and x_{k+1} can be evaluated reusing α (see text). The *arrows* indicate which sub-evidences sum up to a sub-evidence for more than m bin boundaries

classification (GPC) and maximum-entropy binning (MEB), were compared to that of the BBCa. SVMs (Vapnik 1995, 1998) have enjoyed great popularity due to their successes in many applications, e.g. handwritten digit recognition (Cortes and Vapnik 1995) or 3D object recognition (Blanz et al. 1996). For the comparison presented here, *libsvm 2.8*[3] was used. This package also includes a Python script which performs data scaling and model selection for a C-SVM using a radial basis function kernel (for details, see Hsu et al. 2005).

Gaussian processes were originally designed for regression problems (MacKay 2003), for which analytical solutions exist. They can also be used for classification (Williams and Barber 1998), but in this case the required integrations have to be carried out using approximation techniques. We used the Monte-Carlo approach implemented in the package *fbm-2004-11-10*[4] (Neal 1997).

Maximum-entropy binning (MEB) is a standard neural decoding approach (Panzeri and Treves 1996; Rolls et al. 1995). Here, the mapping $f(\vec{w}) \mapsto x$ is chosen such that x can take on only as many different values as y. In sensory neurophysiological experiments, it is common to discretise the respose of a cell (usually the spike count in a given time window) into a number of bins equal to the number of stimuli used. Furthermore, the entropy (Shannon 1948) of x must assume its maximum possible value, i.e. all values of x have to be equally likely. The rationale behind this second requirement might be sought in the data processing inequality (Cover and Joy 1991): if the entropy of x is larger than or equal to the entropy of y, then perfect decoding, i.e. finding a surjective function $g(x) \mapsto y$, could be possible. In MEB, this function is determined by estimating the class-conditional distributions $P(x|y)$ from the available samples, and, given that $P(y)$ is determined by the experimental setup, then converting $P(x|y)$ via Bayes' rule into $P(y|x)$. $g(x) \mapsto y$ is then chosen such that $P(y|x)$ is maximised for all x.

Training and test data were generated by simulating a neuron's response to eight stimuli. The 'strongest' stimulus evoked 60 spikes/s, the 'second strongest' 40 spikes/s, the 'weakest' 15 spikes/s and the remaining stimuli evoked 30 spikes/s. Responses were recorded over a time window of 100 ms, during which the firing rate did not change. The resulting average firing rate was used as the input to the four algorithms. The classification target was the stimulus label. All stimuli were presented equally often. This rather simple scenario was chosen so as to allow for an a-priori determination of the expected classification performance limits.

Our algorithm was trained with a maximum number of $M=50$ bin boundaries. Both a uniform prior over the number of intersections M and a prior $\propto \frac{1}{M^2}$ (i.e. the prior probability for a model is inversely proportional to the computational effort required to evaluate it) were tried, they produced very similar results. For the GPC, prior (hyper)parameters for the covariance function need to be chosen. Various settings were tried and eventually a covariance function comprised of a linear part and a squared-exponential part was chosen. The linear part was given by a Gaussian prior with standard deviation 10 and mean 0. The scale and relevance parameters of the exponential part were Gaussian with mean 0 and variances drawn from broad inverse-gamma distributions with mean 20 and 5, respectively.[5] Changing these prior parameters within two orders of magnitude did not affect the classification performance in any substantial way. As noted above, the SVM package contained Python scripts to perform parameter selection via cross-validation in an automated manner.

The average percentage of correctly classified stimuli as a function of the trials per stimulus (i.e. the number of times a response to a given stimulus appeared in the training set) is shown in Fig. 3. For a given number of trials per stimulus, each classifier was first trained on a training data set, then its performance was evaluated on a test data set containing 100 trials per stimulus. This procedure was repeated on 100 different training/test data sets to allow for an evaluation of means and standard errors. Since all eight stimuli were equally likely, one expects a performance of 12.5% based on this information alone. If the response-generating distributions were known, the optimal performance as predicted by Bayes' rule would be ≈24.5%. All four methods seem to converge towards this value, even though GPC is doing notably worse than the other three. For 1,000 trials per stimulus, BBCa and SVM have virtually reached the theoretical optimum (not shown). However, especially in the neurophysiologically relevant range of only a few available trials per stimulus, the BBCa outperforms the three competitors. This indicates that the BBCa is more suitable for neural response classification than the other three. Moreover, as detailed below, it allows for an exact evaluation of the evidence Eq. (6), which is necessary if subsequent stages of Bayesian inference are to be conducted

[3]Available at http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[4]Available at http://www.gaussianprocess.org.

[5]For details of the possible prior choices, the reader is referred to the extensive documentation of the *fbm-2004-11-10* package.

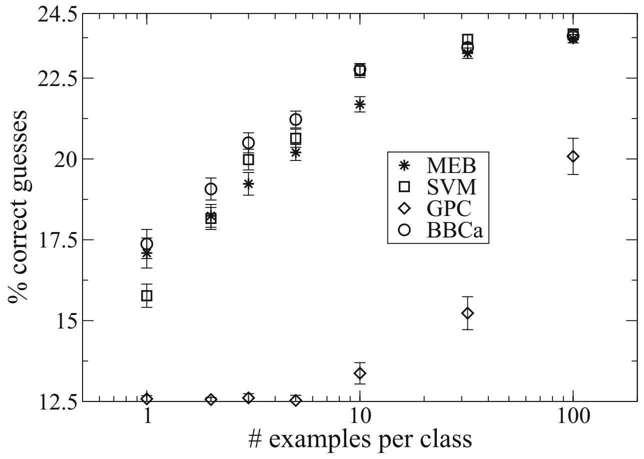

Fig. 3 Comparison of the percentages of correctly classified stimuli as a function of the number of trials per stimulus. *Stars*: MEB, *circles*: BBCa, *squares*: SVM, *diamonds*: GPC. Error bars are standard errors computed from 100 repetitions. The theoretical performance limits are 12.5% (uninformed guess based only on prior knowledge of the stimulus distribution) and ≈24.5% (best possible expected performance if the response generating distributions were known). Especially in the neurophysiologically relevant range of only a few available trials per stimulus, the BBCa outperforms the other methods. For details, see text

without introducing approximation errors. This is an additional advantage which SVM and MEB cannot offer.

7 Application to neural spike data

One of the central questions in neuroscience is how the activities of a large number of individual neurons encode information. One of the best ways to study this question is to measure the responses of single neurons to a range of stimuli, and to determine their tuning properties to certain aspects of these stimuli. Such experimental tests are severely limited by the time available for measuring signals from individual neurons, so the available time needs to be used as efficiently as possible. Recent results have shown that the rapid, continuous presentation of stimuli is highly efficient (Földiák et al. 2004; Keysers et al. 2001), even in high-level visual cortical areas such as STSa. The rapid serial visual presentation (RSVP) method, however, introduces the problem of separating the responses due to each of the rapidly presented stimuli. We need to use response time windows that minimise the effect of interference between subsequently presented stimuli that are thus the most appropriate to linking or classifying the signal with the stimulus that caused this part of the neural response. Using the Bayesian approach, on the one hand, we characterise the temporal properties of the neural response by computing the posterior distribution of the window parameters (i.e., latency and response duration), as well as their expectations and variances. On the other hand, by virtue of Eq. (4) we can evaluate how well the stimulus labels can be predicted from the responses by showing the relationship between response magnitude and expected class probabilities.

We used the BBCa to analyse RSVP spike train data recorded from single cells of area STSa of the visual cortex of monkeys (Keysers et al. 2001). The monkey was presented with a continuous stream of complex natural images which were drawn from a set of eight different images selected from a large image set for each neuron and the resulting firing pattern was recorded. This raw signal was turned into distinct samples, each of which contained the spikes from −250 ms before to 500 ms after the stimulus onset. The temporal resolution of the recording was 1 ms. Time indexes were aligned to the stimulus onset. Here and in the following, a 'dataset' denotes the collection of responses of a cell to a given stimulus set. A 'datapoint' or 'trial' is a stimulus-response tuple. The 'target stimulus' is the stimulus identity to be determined from a given response (i.e. the class label in the classification task). The stimuli were presented multiple times in pseudorandom order. Table 1 shows some of the characteristics of the available data. Stimulus onset asynchrony (SOA) is the time difference between the onsets of two consecutive stimuli. In all datasets used here for analysis, the stimuli were presented without gaps. Thus, SOA was equal to the duration of the stimulus. There was one dataset available for a given cell and SOA, but not all cells were tested at all SOAs. The stimulus set for a given dataset was chosen from 68 complex stimuli prior to recording. For a more detailed description of the recording procedure, see (Keysers 2000; Keysers et al. 2001).

Table 1 Number of cells and trials per stimulus for each of the available stimulus onset asynchronies (SOA)

SOA (ms)	No. of cells	Avg. no. of trials per stimulus
222	40	24
112	42	48
56	40	92
42	28	136
28	40	190
14	28	353

SOA is the time difference between the onsets of two consecutive stimuli. Average trials per stimulus are rounded to the next nearest integer. There was one dataset available per cell and SOA. Each dataset contained responses to a set of 8 stimuli.

A conventional way of decoding such spike trains, i.e. determine the stimuli from the neural resposes, is to count spikes in a time window. Let this window be denoted by $f_0(l, e)$, where l (latency) and e (end) are the first and last time indexes included. The signal extracted from a spike train $s(t_i)$, -250 ms $\leq t_i \leq 500$ ms, $s(t_i) \in \{0; 1\}$ is then

$$x = \frac{\sum_{t_i=l}^{e} s(t_i)}{e - l + 1} \tag{24}$$

i.e. the average firing rate. To find the expected window by means of Bayesian analysis, it is necessary to evaluate the posterior

$$P(f_0(l, e)|D) = \frac{P(D|f_0(l, e))P(f_0(l, e))}{\sum_l \sum_{e \geq l} P(D|f_0(l, e))P(f_0(l, e))} \tag{25}$$

where D is the set of recorded spike trains. The summation in the denominator runs over all start/end times which one chooses to include. All possible values of $l \geq l_{\min}$ and $e \leq e_{\max}$ were assumed to be equally likely prior to observing data, with the restriction $l \leq e$.

First, $P(D|f_0(l, e))$ must be evaluated. To do so, all spiketrains in the dataset were subjected to the transformation [Eq. (24)], thus yielding D_f. This transformed dataset was then fed into the BBCa. Hence, the evidences $P(D|m, f_0(l, e))$ became available. Assuming $P(m, f_0(l, e)) = P(m)P(f_0(l, e))$, i.e. the prior over the number of bins is independent of the prior over the window parameters, one can then write

$$\begin{aligned} P(D|f_0(l, e)) &= \sum_{m=M_0}^{M_1} P(D, m|f_0(l, e)) \\ &= \sum_{m=M_0}^{M_1} P(D_f|m)P(m) \end{aligned} \tag{26}$$

where M_0 and M_1 are the minimum and maximum number of bin boundaries, respectively. In this application, it was sufficient to choose $M_0 = 0$ and $M_1 = 10$, because the maximum of $P(D|m, f_0(l, e))$ was in most cases between $m = 1$ and $m = 7$. The evidence $P(D)$ is obtained by summing the numerator over all l, e and m.

With this posterior, one can evaluate the expectations

$$E[l] = \sum_l \sum_{e \geq l} P(f_0(l, e)|D)l \tag{27}$$

$$\mathrm{Var}[l] = \sum_l \sum_{e \geq l} P(f_0(l, e)|D)l^2 - E[l]^2 \tag{28}$$

and likewise for e and $e - l + 1$. This yields the expected latency, window end and window width, along with their variances.

7.1 Multiple datasets, joint and marginal expectations

In neurophysiological experiments, the number of stimulus repetitions that can be achieved is often quite limited. Thus, pooling data from different runs, possibly recorded in response to different stimulus sets, and perhaps even from different cells, is desirable. If the parameters to be estimated can be expected to have similar values across datasets, then Eq. (26) can be replaced by

$$P(\{D^j\}|f_0(l, e)) = \prod_j \sum_{m_j=M_0}^{M_1} P(D_f{}^j|m_j)P(m_j) \tag{29}$$

where j runs over all datasets, assuming that the datasets were drawn independently. This allows for the computation of expectations of the joint distribution which describes the population response.

When pooling data from the same cell recorded at different presentation rates, it would seem sensible to assume that the latencies should be more alike than the response durations. Thus, one would compute a single latency by marginalisation:

$$\begin{aligned} P(\{D^j\}, f_0(l, .)) = \prod_j \sum_{e_j \geq l} \sum_{m_j=M_0}^{M_1} P(D^j|m_j, f_0(l, e_j)) \\ \times P(m_j)P(f_0(l, e_j)) \end{aligned} \tag{30}$$

$$P(f_0(l, .)|\{D^j\}) = \frac{P(\{D^j\}, f_0(l, .))}{\sum_l P(\{D^j\}, f_0(l, .))} \tag{31}$$

where the denominator of the r.h.s. of Eq. (31) is the evidence of this hypothesis (i.e. all datasets have the same latency but different response durations). Thus

$$E[l] = \sum_l P(f_0(l, .)|\{D^j\})l \tag{32}$$

Likewise, an evaluation of the overall response duration can be performed by marginalising over the latencies.

8 Results on artificial data

The BBCa was tested on artificial data first. Those were generated by a simulated neuron having a response latency of 100 ms and a response duration of 110 ms. The firing probability, i.e. the probability of observing an event in a given time bin, was assumed to be uniform, being 12 spikes/s for the 'strongest' stimulus, 10 spikes/s for the 'second strongest', 2 spikes/s for the 'weakest' and 5 spikes/s for the rest. These values resemble those found in the real data for a moderately weakly responding cell. Therefore, the distribution of the spike count during the response followed a binominal (or approximately a Poisson) distribution. Firing rates before the response latency and after the response were 5 spikes/s. Fig. 4, top, shows the results.

Even for a dataset containing as little as ten trials per stimulus (i.e. the number of times each stimulus was presented), the expected window boundaries are close to their real values. However, their standard deviations are still high. As the number of trials per stimulus grows, the standard deviations decrease and the expected start/end positions move closer to their real values. At 33 trials per stimulus, the standard deviations are small enough to yield useful results in neurophysiological experiments. From 100 trials per stimulus onward, the correct window parameters are determined by the BBCa almost with certainty.

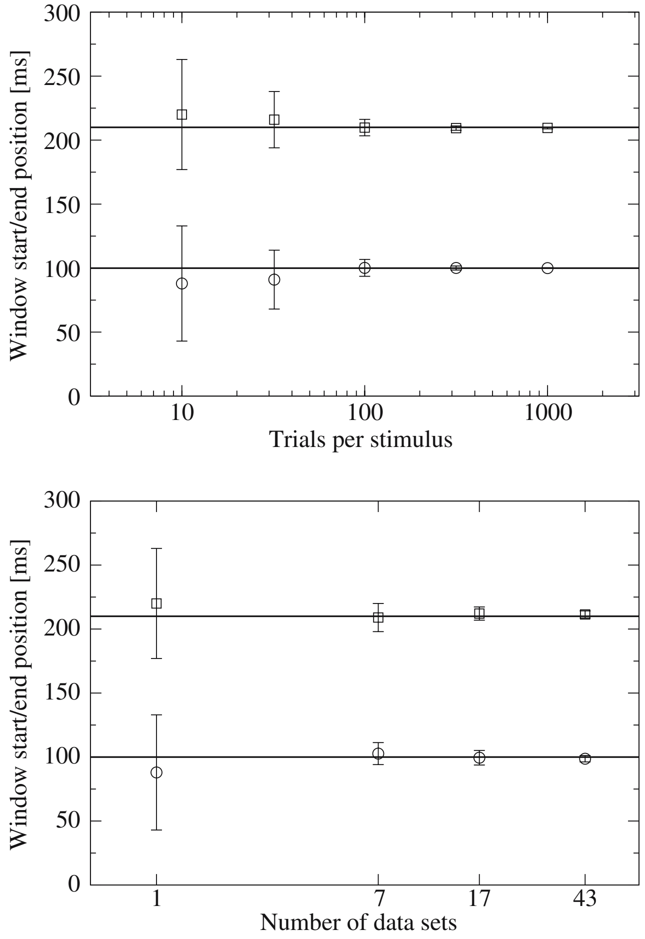

Fig. 4 *Top*: expected window start (*circles*) and end (*squares*) as a function of the number of trials per stimulus. *Bottom*: expected window start (*circles*) and end (*squares*) as a function of the number of datasets. Each dataset contained ten trials per stimulus. Error bars are ±1 standard deviation. Averages computed over 100 runs

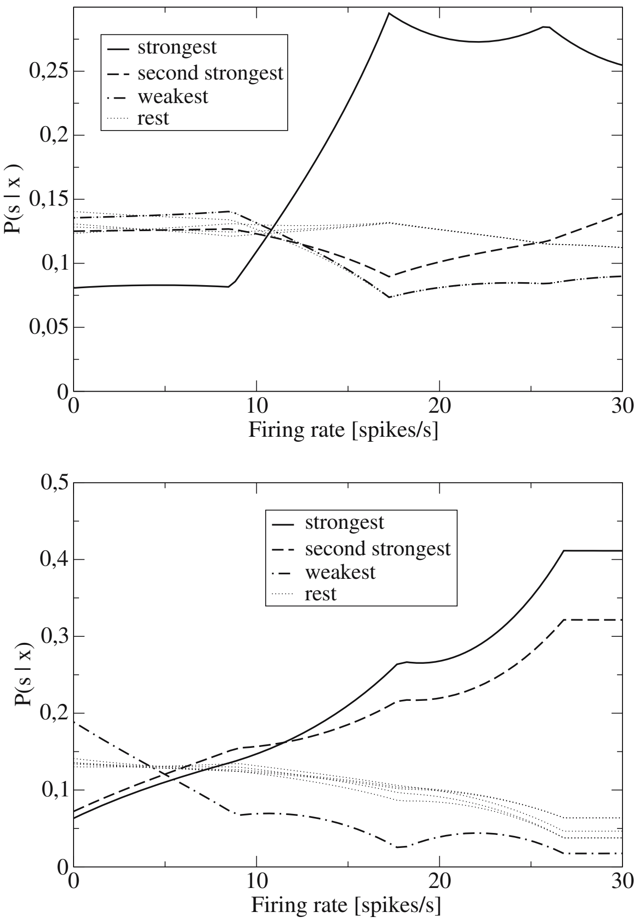

Fig. 5 Expected classification performances for two datasets of different sizes. *Top*: 10 trials/stimulus. The only distinction possible is that between ‘strongest’ and ‘weakest’. *Bottom*: 1,000 trials/stimulus. Here, 3 classes (‘strongest’, ‘second strongest’ and ‘weakest’) can be separated

The results for different numbers of datasets are depicted in Fig. 4, bottom. Each dataset contained ten trials per stimulus. The behaviour is similar to that observed in Fig. 4, top. Performance at 7 datasets with 10 trials per stimulus each is comparable to 1 dataset with 100 trial per stimulus. It appears that the total number of available trials (i.e. number of trials per stimulus in a dataset × number of datasets) is the determining factor for the quality of the results.

Classification performance for two datasets of different sizes is plotted in Fig. 5. If a stimulus was to be identified by the response it evoked, one would pick the stimulus that maximises $P(s|x)$, where s is the stimulus and x is the response. At ten trials/stimulus, the only distinction possible is that between ‘strongest’ ($x > 10.8$) and ‘weakest’. At 1,000 trials per stimulus, 3 classes can be separated: ‘weakest’ ($x < 4.3$), ‘second strongest’ ($6.0 \leq x < 11.3$) and ‘strongest’ ($x \geq 11.3$). The range $4.3 \leq x < 6.0$ would then be assigned to 'rest'. Note that those boundaries are in good agreement with the firing rates that were used to generate the responses.

9 RSVP results

9.1 How similar are cells?

The available data contained measurements for six different stimulus onset asynchronies (SOA): 14, 28, 42, 56, 112 and 222 ms. As explained above, here SOA is the time interval between the onsets of two consecutive stimuli. In the following, the terms 'information latency' (IL) and 'information response duration' (IRD) refer to the time indexes of the window start and window length as determined by the BBCa. The term 'response latency' (RL) denotes the time index at which a response was detected by a method described in hypothesis H2 (see below). Six hypotheses were compared:

H1. Joint: for a given SOA, all cells have the same IL and IRD. While previous results (see e.g. Keysers et al. 2001) indicate that this assumption is likely to be wrong, it served as a 'null hypothesis' against which the other hypotheses were compared.

H2. IL=RL: this is a popular hypothesis in neurophysiology. The response latency was determined on a dataset by dataset basis by a method described in Keysers et al. (2001): The spike train is smoothed by a Gaussian of width 10 ms to yield a spike density function. Its mean μ and standard deviation σ are computed in a time window of 250 ms directly prior to the stimulus onset. The RL is defined as the first 1 ms time bin after which the spike density function exceeds $\mu + 2.58\sigma$ for at least 25 ms. Moreover, the response duration was taken to be equal to the stimulus duration.

H3. Latency: for a given SOA, all cells have the same IL, but possibly different IRDs. This assumption allows for the computation of an overall IL as a function of the presentation rate.

H4. Window length: for a given SOA, all cells have the same IRD, but possibly different ILs. This assumption allows for the computation of an overall IRD as a function of the SOA.

H5. Single: for a given SOA, each cell has a different IL and IRD.

H6. Latency per cell: each cell has an IL, which does not change with the presentation rate. The IRD may change with the presentation rate. This hypothesis is similar to the one used in Keysers et al. (2001) for the determination of the latency of each cell.

For H2, only the datasets with SOAs 42–222 ms were used, since the faster presentation rates do in most cases not yield enough signal to allow for a determination of the RL. Thus, two sets of comparisons were performed: the first included H1, H2, H3, H4 and H5 (left half of Table 2), the second H1, H3, H4, H5 and H6 (right half of Table 2). H1 was used as the point of reference for all others, i.e.

$$\log_{10}(\text{evidence}) \text{ gain of HX} = \log_{10}(\text{evidence of HX}) - \log_{10}(\text{evidence of H1}) \tag{33}$$

As expected, H1 is the most unlikely one. IL = RL already results in a substantial evidence gain (being $\approx 10^{47}$ times more probable). However, compared to the remaining hypotheses, it can still safely be discarded, indicating that this way of information extraction will yield suboptimal results.

Table 2 Evidence comparisons for the different hypotheses described in the text. $\log_{10}$(evidence) gains are computed w.r.t. the evidence of H1

42–222 ms		14–222 ms	
Hypothesis	$\log_{10}$(evidence) gain	Hypothesis	$\log_{10}$(evidence) gain
H1	0.00	H1	0.00
H2	47.5	H3	199.3
H3	167.6	H4	400.5
H4	301.9	H6	456.7
H5	367.1	H5	494.4

For the evidence comparisons in the left half of the table, only recordings with SOAs between 42–222 ms were used, because the shorter presentation rates do in most cases not yield enough signal to allow for a determination of the RL. For the comparisons in the right half, recordings at all available SOAs were included

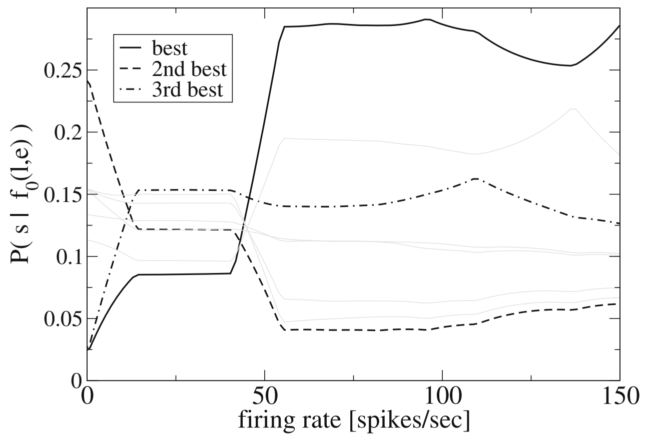

Fig. 6 Expected classification performance for cell 144.4. Three stimuli can be separated. Stimulus ranking was done by $P(s|\,f_0(l,e))$, i.e. the best stimulus is the one which can be identified with the greatest certainty given the response computed from the spike train via $f_0(l,e)$. For details, see text

Comparing H3 and H4 shows that the IRDs seem to be more alike than the ILs, indicating that the latency depends more strongly on the cell than the response duration does. This is consistent with H6 being more probable than either of them. Nevertheless, as noted above, they serve to compute the expected ILs and IRDs across all cells.

H5, despite its additional degrees of freedom (one IL and IRD per cell and SOA), appears to provide the best explanation of the data. In other words, both IL and IRD depend strongly on both the cell and the SOA. This result should not be understood to mean that averaging over cells will generally yield meaningless results, but rather that when such averaging is performed (e.g. to compute information flows), it is advisable to compute the IL and IRD anew for each cell and SOA.

Figure 6 shows the expected classification performance for a single cell at SOA 56 ms. Three stimuli can be separated. Ranking was done by expected certainty of stimulus identification, i.e. by the probability $P(s|\,f_0(l,e))$ that stimulus s had been presented given the response computed from the spike train via $f_0(l,e)$. Thus, the 'second best' stimulus is not the one with the second strongest response, but rather the one with the weakest [this way of ranking stimuli is different from (Keysers et al. 2001), where stimuli were ordered by response strength]. This begets the question: is information transmitted through not firing as well as through firing, and if so, how much?

Moreover, note that even the 'best' stimulus can only be classified correctly with a probability of at most ≈ 0.3. While this allows for a decision with greater certainty than a decision based solely on prior knowledge ($p = 0.125$ for eight stimuli), it is still fairly low. In a 'best' vs 'rest' decision task, one would therefore always choose 'rest'. Thus, additional information is required if the stimulus is to be identified. That that is possible at SOA 56 ms has been shown in Keysers et al. (2001) via human psychophysical experiments using the same stimulus set as that employed for the single-cell recordings which yielded the data for the analysis presented here.

9.2 Latencies and response durations

The overall ILs and IRDs computed via H3 and H4, respectively, are shown in Fig. 7. In accordance with previous results on the same data (Keysers et al. 2001), the IRD increases with the SOA. For SOAs greater than ≈50 ms, the IRD was found to be shorter than the

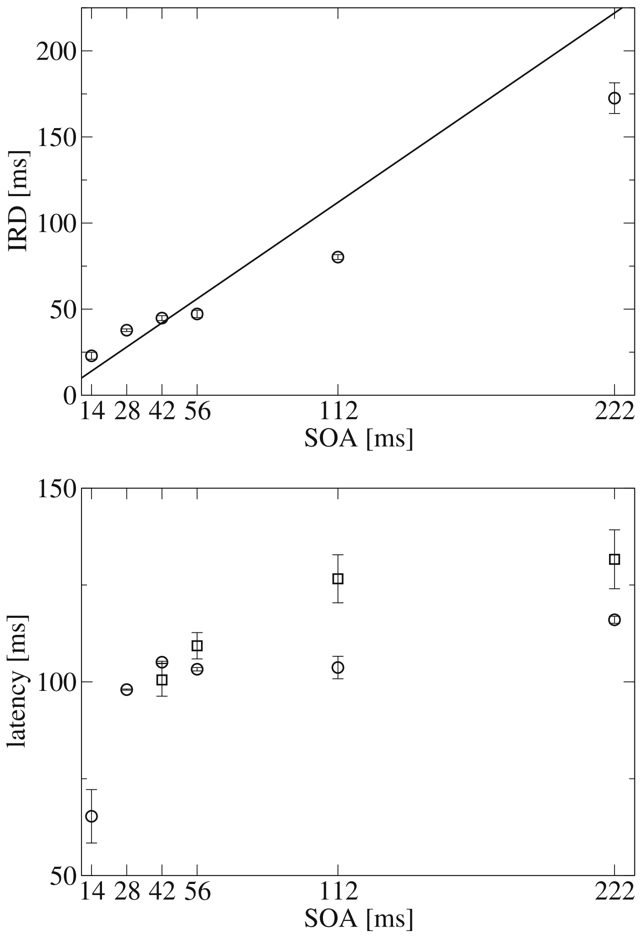

Fig. 7 *Top*: Expected IRDs ± one standard deviation, averaged over all cells. The *solid line* marks the SOA. IRD is shorter than the SOA when the latter is ≥56 ms, and longer for SOAs ≤46 ms. *Bottom*: Expected ILs ± one standard deviation (*circles*) and average RLs (*squares*). Shorter stimuli appear to result in a quicker information flow onset. For 42 and 14 ms SOA, data were available only for a fraction of the cells that were tested at the other SOAs

SOA, this behaviour is reversed for shorter SOAs. This appears at first contradictory to (Keysers et al. 2001), where the response was reported to outlast the SOA by ≈60 ms. That might, however, be due to a difference in method: in the aforementioned study, response offset was defined as the start of the first 30 ms time window in which all 1 ms bins failed a t-test ($p > 0.05$) performed on the spike density functions of the best stimulus and all other stimuli together. In contrast, the approach presented here calculates the expected response length by averaging over all possible windows weighted by their classification performance. Therefore, it will tend to exclude parts of the response which, while they may still contain some information about the stimulus, will decrease the signal-to-noise ratio compared to the 'better' parts available. Thus, the windows can be expected to be somewhat shorter.

The IL seems to increase with the SOA, too. For SOA = 42 and 14 ms recordings were made only on a subset of the available cells, this might offer an explanation for the slight increase of the IL at 41 ms SOA compared to 56 ms SOA. Nevertheless, the overall trend of the IL is still clearly discernible. The large standard deviation of the IL at 14 ms SOA also indicates that at this very high presentation rate, classification information in the response becomes increasingly hard to localise.

10 Conclusion

10.1 Algorithm

The BBCa exactly computes evidences and expected classification probabilities with a computational effort of $O(K^2M^2)$, where K is the number of data points and $M+1$ is the number of bins. This is significantly faster than a naïve approach of simply trying all possible permutations of the bin boundaries between the data points, which would take $O(K^M)$. This acceleration is accomplished by reusing intermediate results in a way similar to dynamic programming (Bertsekas 2000) or the sum-product algorithm (Kschischang et al. 2001). Moreover, it yields the evidence of the data given the considered model class, which is here quantified by M, the number of bin boundaries. This evidence, which is required if further stages of inference are to be performed, is usually not available when using methods that compute the class-conditional probabilities first and convert them afterwards into the probabilities of the class labels via Bayes' theorem.

Note that the expected probabilities [Eq. (4)] are, by virtue of Eqs. (6) and (21) polynomials of order M in x: for example, Eq. (23) is quadratic. The coefficients of these polynomials change whenever x passes over a data point, in such a way that $P(y|x)$ remains continuous. Thus, one could also look at the BBCa as a form of piecewise Bayesian polynomial interpolation.

As mentioned above, finding a suitable mapping from the feature vector onto a scalar, $f(\vec{w}) \mapsto x$, is generally not trivial. Most likely, exact inference of $f(\vec{w})$ will not be possible, and thus an approximation scheme needs to be employed. No matter which technique one decides to try, it will usually be necessary to evaluate a quantity proportional to

$$p(f(\vec{w})|D) \tag{34}$$

i.e. Eq. (25) with $f_0(l, e)$ replaced by $f(\vec{w})$. The denominator of Eq. (25) will probably be intractable, but the numerator is readily accessible, once a prior over the possible functions has been chosen. Then, a Monte-Carlo technique can be applied to determine e.g. the expected $f(\vec{w})$.

Another possibility, which might speed up the inference process, is to find the maximum of Eq. (34) w.r.t. $f(\vec{w})$ via a gradient-based technique. Lets assume $f(\vec{w}) = f(\vec{w}, \vec{\theta})$, where $\vec{\theta}$ is a vector of parameters that governs the exact form of $f(\vec{w}, \vec{\theta})$. Since Eq. (6) is a polynomial in the x_k, its gradient w.r.t. the x_k can be evaluated by an algorithm similar to the BBCa. Thus, using the chain rule,

$$\nabla_{\vec{\theta}} \log(p(f(\vec{w}, \vec{\theta})|D)) = \sum_{k=0}^{K-1} \frac{\partial \log(P(D|f(\vec{w}, \vec{\theta})))}{\partial x_k} \times \nabla_{\vec{\theta}} f(\vec{w}_k, \vec{\theta}) + \nabla_{\vec{\theta}} \log(p(\vec{\theta})) \tag{35}$$

$\frac{\partial \log(P(D|f(\vec{w},\vec{\theta})))}{\partial x_k}$ can, by virtue of Eq. (26), be computed via the derivative of Eq. (6) w.r.t. x_k. $p(\vec{\theta})$ is the prior of $\vec{\theta}$. $\nabla_{\vec{\theta}} f(\vec{w}_k, \vec{\theta})$ depends on the exact functional form of $f(\vec{w}, \vec{\theta})$. Assuming, for instance, that it is a feed-forward neural network with a single output, then $\vec{w}_k$ would be the vector of inputs for the k-th example, x_k the resulting output and $\vec{\theta}$ stands for the weights. $\nabla_{\vec{\theta}} f(\vec{w}_k, \vec{\theta})$ could then be computed by some variant of the backpropagation algorithm (Rumelhart et al. 1986), and thus gradient ascent on $\nabla_{\vec{\theta}} \log(p(f(\vec{w}, \vec{\theta})|D))$ becomes feasible. Once the weights maximising the posterior have been determined, refinements are of course possible, such as using a Laplace approximation for the weight posterior, or Monte-Carlo techniques. The important point is that, since a feedforward network with sigmoid transfer functions can model any continuous mapping onto a scalar to any desired degree of accuracy given that the number of hidden units is allowed to grow (Cybenko 1989), one might hope that this approach could in principle solve any classification task for which $f(\vec{w}, \vec{\theta})$ can be reasonably well approximated by a continuous function.

The applicability of the BBCa for the analysis of neural spike train data has been demonstrated. It yields useful results even when only ≈10–100 trials per stimulus are available, which can usually be accomplished in recordings of single cells. Thus, analyses can be conducted not only on a population level, but also on a cell by cell basis.

10.2 STSa neuron populations adapt their processing speed to the presentation rate

In Section 9.2, we described a quasi-monotonic increase of IL and IRD with the SOA. This may suggest that the visual system speeds up the processing of stimuli in the fast presentation (low SOA) conditions, i.e. the shorter the presentation time, the faster the information flow onset. Alternatively, it might simply be due to an increased overlap of the neural responses in the fast conditions. A third possible explanation may involve feedback. We will argue against these alternative explanation, which are not supported by our analysis.

If response duration were longer than SOA then responses to subsequent stimuli would overlap in time. The overlapping response from the randomised previous stimulus would be noise for the task of determining the identity of the current stimulus. For small overlaps, this noise would most affect the early part of the response. An optimal window could then be expected to start after the end of such interference thus the IL should increase as SOA decreases. This is not what we observe in our results (see Section 9.2).

If response overlap explained the IL shift then one would expect to find a gradual increase of stimulus information before the IL. In contrast, we observe a sudden onset. In Fig. 8 the logarithm of the evidence [Eq. (26)] divided by the number of data points is plotted as a function of $e - \mathrm{IL}$, where $e - l = 10$ ms, i.e. the log evidence for a 10 ms sliding window (thin dotted lines) and also its running average over 11 ms (thick lines). This quantity will be high if good classification is possible, and low otherwise. Thus its value indicates how much information the spike count carries about the stimulus. The log evidence values have been aligned so that their average in the interval -250 ms $\leq e - \mathrm{IL} \leq -150$ ms is 0. The responses in this time interval should contain no information about the stimulus and can thus provide baseline values for the curves. Note that the rising slopes which indicate the onset of the information flow are almost perfectly aligned with each other for SOAs 222, 112 and 56 ms. Since the IL at SOA 222 ms is ≈13 ms larger than those at SOA 112 and 56 ms, this indicates that the cell population really begins transmitting later at the longest SOA. The situation is similar for the three shorter SOAs. Even though the rising slopes for 42 and 28 ms are not perfectly aligned, their temporal offset (≈5.5 ms) is smaller than ≈7 ms, which is the difference in their ILs. This is true to an even stronger degree for the shortest SOA: IL(SOA = 28 ms) − IL(SOA = 14 ms) ≈ 33 ms, yet their rising slopes are within a few ms of each other.

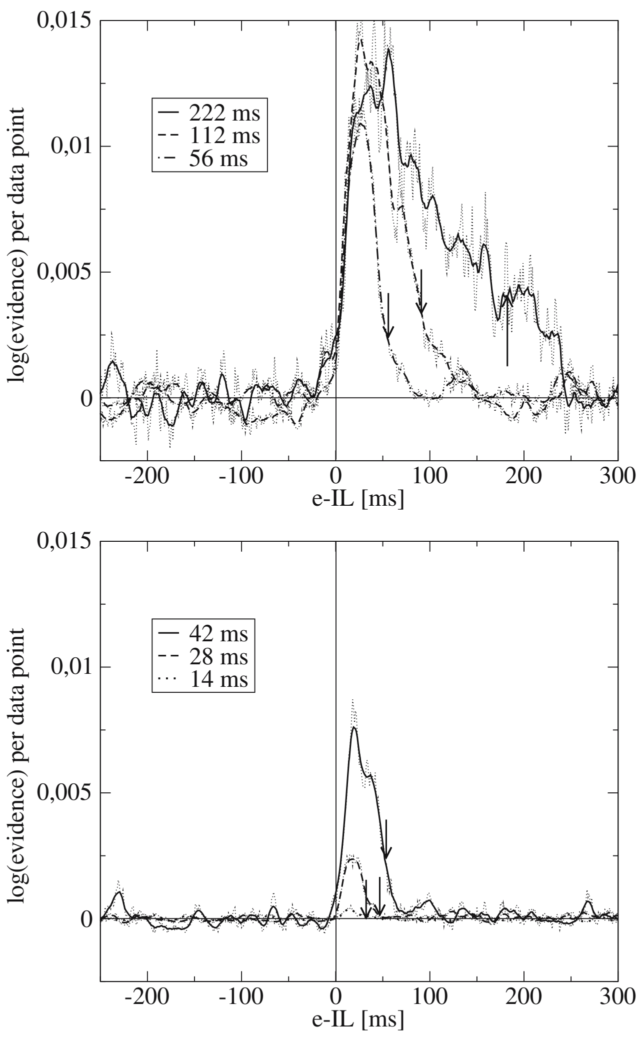

Fig. 8 $\log(P(D_f | f_0(e - 10\text{ ms}, e))$ per data point, i.e. log evidence per data point in a sliding time window of length 10 ms (*thin dotted lines*) and their running averages in an 11 ms window (*thick lines*). Values are aligned so that the average for $e - IL < -150$ ms are zero. There is no indication of stimulus-related information when e < IL. The arrows indicate the end of the optimal time window for stimulus discrimination. Some information seems to be transmitted after that, but including this latter part of the response would reduce classification performance

We will now examine the feedback hypothesis. Feedback connections both within and between areas is a prominent feature of cortical organisation. Experiments with artificial neural networks (Endres and Földiák 1999; Endres 2006; Földiák 1990; Harpur and Prager 2000) show that lateral feedback connections can give rise to sparse, almost factorial neural codes. Such codes facilitate pattern recognition and are consistent with the observed properties of neurons in primary visual cortex. Feedback processing must have implications for the time course of neural response, as synaptic delays limit the minimal time for convergence. It is unlikely that the later processing stages would wait until the earlier ones have converged, i.e. there would be a signal arriving in higher areas even though the lower areas are still busy making sense of their input. This early signal could convey some information about the stimulus but not as much as the later part which would become available if the stimulus was presented for long enough. There is no evidence for this type of signal improvement in Fig. 8. This result is consistent with Oram and Perrett (1992); Thorpe and Imbert (1989), who argued that under these conditions information flow from retinal output to STSa has to be predominantly feedforward in order to achieve the observed short latencies.

As noted above, IRD $<$ SOA if SOA $> \approx 50$ ms. Thus, for longer stimuli, the visual system should be able to separate the responses to successive stimuli, because IRD is the duration of the response needed for best stimulus discrimination. This is no longer the case for the shorter SOAs: here responses to stimuli will begin to overlap, and thus optimal classification performance can no longer be attained.

One possible mechanism that might explain the observed variations in IL and IRD is threshold adaptation or, to the same end, residual activation: suppose the firing threshold of the cell increased with SOA. Then it would begin to fire sooner as the SOA decreases, and also longer w.r.t. SOA. It is currently unclear what might bring about such an adaptation. But if one assumes that the cell behaves to some degree like a 'leaky integrator', then in the shorter SOA conditions, when it is exposed to a 'good' stimulus, there might still be some activation left from its last presentation. This activation would have 'leaked' out of the cell had the SOA been longer.

As shown in Fig. 6, the probability of correctly identifying the stimulus given the response computed from the spike train via $f_0(l, e)$ does not exceed 0.3 (this result is of course conditioned on the cell and the stimulus set). The cell from whose responses this classification graph was computed is a fairly average specimen of the population that was sampled. It is thus evident, that even for a stimulus set as limited as the one used in this experiment, more information is required to identify the stimulus with reasonable certainty (say, 99%). This information would become available if one observed not the activity of a single cell, but that of a population. As demonstrated in Földiák (1993); Oram et al. (1998), combining the signals of a few cells via Bayes' theorem on the assumption that their responses (given the stimulus) are conditionally independent can yield the desired increase of certainty.

Acknowledgements We would like to thank Dengke Xiao and David Perrett for making the data available to us. We would also like to thank Mike Oram and Johannes Schindelin for useful discussions and the anonymous reviewers for helpful criticisms of this manuscript. D. Endres was supported for a part of this project by the MRC Special Training Fellowship in Bioinformatics. P. Földiák was supported for parts of this project by the BBSRC and the Albert Szent-Györgyi Fellowship, Hungary.

References

Bertsekas, D. P. (2000). *Dynamic programming and optimal control*. Athena Scientific.

Blanz, V., Schölkopf, B., Bülthoff, H. C. B. V. V., & Vetter, T. (1996). Comparison of view-based object recognition algorithms using realistic 3D models. In C. von der Malsburg, W. von der Seelen, J. C. Vorbrüggen & B. Sendhoff (Eds.), *Artificial neural networks—ICANN96* (pp. 251–256). Berlin: Springer.

Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning*, *20*, 273–297.

Cover, T. M., & Joy, T. A. (1991). *Elements of information theory*. New York: Wiley.

Cybenko, G. (1989). Approximations by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, *2*, 304–314.

Endres, D., & Földiák, P. (1999). Quadratic programming for learning sparse codes. In *Proceedings of the ninth international conference on artificial neural networks (ICANN99), IEE Conference Publication No. 470* (pp. 593–596). London. Institution of Electrical Engineers.

Endres, D. M. (2006). Bayesian and information-theoretic tools for neuroscience. Ph.D. diss., University of St. Andrews, Scotland, UK.

Földiák, P. (1990). Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics*, *64*, 165–170.

Földiák, P. (1993). The 'Ideal Homunculus': Statistical inference from neural population responses. In F. Eeckman & J. Bower (Eds.), *Computation and neural systems* (pp. 55–60). Norwell, MA: Kluwer.

Földiák, P., Xiao, D., Keysers, C., Edwards, R., & Perrett, D. I. (2004). Rapid serial visual presentation for the determination of neural selectivity in area STSa. *Progress in Brain Research*, *144*, 107–116.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (1995). *Bayesian data Analysis*. Boca Raton: Chapman & Hall.

Harpur, G. F., & Prager, R. W. (2000). Experiments with low-entropy neural networks. In R. Baddeley, P. Hancock & P. Földiák (Eds.), *Information theory and the brain*, Chap. 5 (pp. 84–100). New York: Cambridge University Press.

Hsu, C., Chang, C., & Lin, C. (2005). *A practical guide to support vector classification*. Retrieved at http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

Hutter, M. (2002). Distribution of mutual information. In *Advances in neural information processing systems 14* (pp. 339–406). Cambridge, MA: MIT Press.

Jordan, M., Ghahramani, Z., Jaakkola, T., & Saul, L. (1999). An introduction to variational methods for graphical models. *Machine Learning*, *37*, 183–233

Keysers, C. (2000). *The speed of sight.* Ph.D. diss., School of Psychology, University of St. Andrews, UK

Keysers, C., Xiao, D., Földiák, P., & Perrett, D. I. (2001). The speed of sight. *Journal of Cognitive Neuroscience*, *13*, 90–101.

Kschischang, F., Frey, B., & Loeliger, H. A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, *47*, 498–519.

MacKay, D. J. C. (2003). *Information theory, inference and learning algorithms*. New York: Cambridge University Press.

Neal, R. M. (1997). *Monte Carlo implementation of Gaussian process models for Bayesian regression and classification.* Technical report 9702, Dept. of Computer Science, University of Toronto.

Neal, R. (1996). *Lecture notes in statistics: Bayesian learning for neural networks*, Vol. 118. New York: Springer.

Oram, M. W., Földiák, P., Perrett, D. I., & Sengpiel, F. (1998). The 'Ideal Homunculus': Decoding neural population signals. *Trends in Neuroscience*, *21*, 259–265.

Oram, M. W., & Perrett, D. I. (1992). Time course of neural responses discriminating different views of the face and head. *Journal of Neurophysiology*, *68*, 70–84.

Panzeri, S., & Treves, A. (1996). Analytical estimates of limited sampling biases in different information measures. *Network: Computation in Neural Systems*, *7*, 87–107.

Rolls, E., Critchley, H., & Treves, A. (1995). The representation of olfactory information in the primate orbitofrontal cortex. *Journal of Neurophysiology*, *75*, 1982–1996.

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning internal representations by error propagation. In D. Rumelhart, J. McClelland & the PDP research group (Eds.), *Parallel distributed processing: explorations in the microstructure of cognition* (pp. 318–362). Cambridge, MA: MIT Press.

Shannon, C. E. (1948). The mathematical theory of communication. *Bell Systems Technical Journal*, *27*, 379–423, 623–656.

Thorpe, S., & Imbert, M. (1989). Biological constraints on connectionist modelling. In R. Pfeifer, Z. Schreter & F. Fogelman-Souli'e (Eds.) *Connectionism in perspective* (pp. 63–93). Elsevier.

Vapnik, V. (1995). *The nature of statistical learning theory*. New York: Springer.

Vapnik, V. (1998). *Statistical learning theory*. New York: Wiley.

Williams, C. K. I., & Barber, D. (1998). Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*, 1342–1351.